Neuralnet Collocation Method for Solving Differential Equations (Computation mechanics and domain decomposition methods)

AUTHOR(S):

Takeda, Tatsuoki; Fukuhara, Makoto; Liaqat, Ali

数理解析研究所講究録
1129巻 2000年 115-128

Neuralnet Collocation Method for Solving Differential Equations

Tatsuoki TAKEDA, Makoto FUKUHARA, Ali LIAQAT
Department of Information Mathematics and Computer Science
The University of Electro-Communications

電気通信大学　竹田辰興、福原誠、リアカト・アリ

Abstract

Collocation method for solving differential equations by using a multi-layer neural network is described. Possible applications of the method are investigated by considering the distinctive features of the method. Data assimillation is one of promising application fields of this method.

1. Introduction

1.1 Neural network

In this article we present an application of a feed-forward multi-layer neural network (neuralnet) as a solution method of a differential equation. The feed-forward multi-layer neural network is a system composed of (1) simple processor units (PUs) placed in layers and (2) connections between the PUs of adjacent layers (Fig.1). Data flow from the input layer to the output layer through the hidden (intermediate) layers and the connections to which weights determined by a training process are assigned. At each PU of the hidden layers and the output layer weighted sum of the incoming data is calculated and the result is nonlinearly transformed by an activation function and transferred to the next layer. Usually some kind of sigmoid functions is used as the activation function. But at the output layer often the nonlinear transformation is omitted. The neural network is usually trained as follows. (1) A lot of datasets composed of input data and corresponding output data (supervisor data) are prepared. (2) Output data are calculated by entering the input data to the input PUs according to the above data flow. (3) Sum of squared errors of the output data from the supervisor data is calculated. (4) Weights assigned to the connections are adjusted so that the above squared sum is minimized. The most commonly used algorithm to this process is the gradient method and, in the case of the feed-forward multi-layer neural network the error back

propagation method based on the gradient method is implemented efficiently.

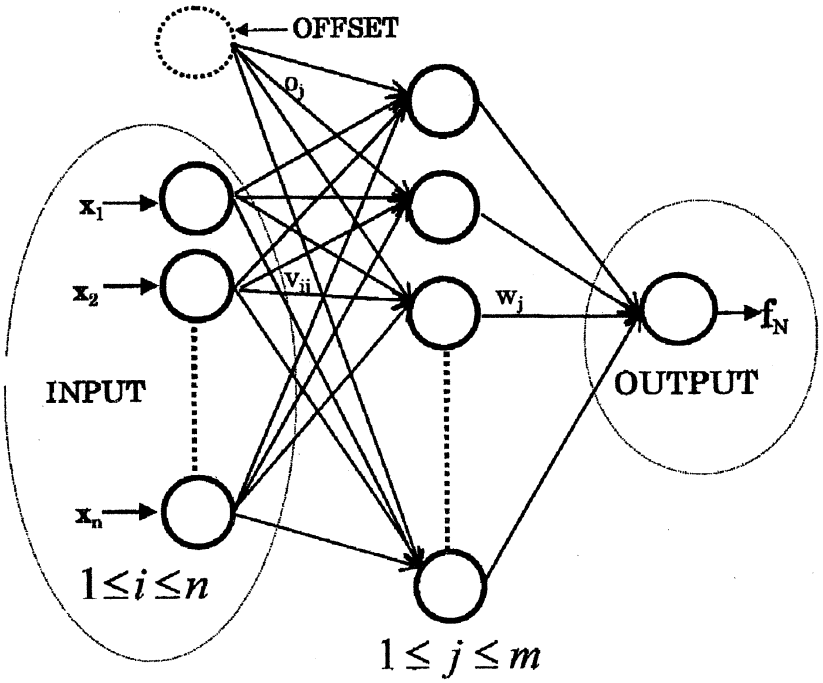


Fig.1 A schematic diagram of a three-layered neural network

1.2 Features of a neural network

The feed-forward multi-layer neural network is characterized by keywords of training, mapping, smoothing, and interpolation. In relation with the keywords, we make use of the following distinctive features of the neural network for our purpose.

(1) Training: Optimization of the set of the weights with respect to an appropriate object function is carried out. The object function of the neural network is represented as

$$E = \sum_{pattern} F\left(\vec{f}(\vec{x}_{pattern}, \vec{p})\right)$$

where $\vec{x}_{pattern}$ is an input data vector for the training and $\vec{p}$ is a vector representing the weights. For the feed-forward multi-layer neural network the following type of object functions is usually employed.

$$F\left(\vec{f}(\vec{x}_{pattern}, \vec{p})\right) = \left[\vec{f}(\vec{x}_{pattern}, \vec{p}) - \hat{\vec{f}}_{pattern}\right]^2$$

Usually a sum of squared errors of output data from the prepared supervisor data is used for the object function as above, but it should be

remarked that there are other possibilities for choice of the object function. The training by the error back propagation method is carried out as

$$\vec{p} \Rightarrow \vec{p} - \alpha \frac{\partial E}{\partial \vec{p}}$$

(2) Mapping: By the neural network mapping from the input space to the output space good approximation can be attained in comparison with a usual orthogonal function expansion. It is because in this method not only expansion coefficients but also the basis functions of the expansion themselves are optimized.

(3) Smoothing and interpolation: These are carried out in the meaning of the least square fitting. It is important that these are always associated with the above neural network mapping processes.

1.3 Object function and aims of our study

In our study we applied the feed-forward multi-layer neural network to the solution method of differential equations. Milligen et al., and Lagaris et al., have proposed the method and demonstrated successful results by the method [1,2]. This is a kind of the collocation methods, i.e., the whole procedure is described as follows.

(1) From the computational domain Ω a subdomain $\hat{\Omega}$ composed of collocation points is constructed.

(2) The coordinates of a collocation point are regarded as a pattern for the input data of the neural network and a solution value corresponding to the coordinates is regarded as the output data of the neural network.

(3) The residual of the differential equation is calculated from the output data and the squared residual is used as the object function. In this process it should be noted that the residual is expressed analytically because the output data are the analytical functions of the input variables with the weight variables as parameters.

Difference of the methods by Milligen et al., and Lagaris et al., is in the treatment of the boundary/initial conditions. In the method by Milligen et al., these conditions are treated as penalty terms in the object functions. These conditions are, therefore, satisfied only approximately. In the method by Lagaris et al., on the other hand, these conditions are exactly satisfied by employing appropriate form factors multiplied to the neural

network output variables. However, it is rather difficult or impossible to find form factors appropriate for a given problem.

Moreover, the solution method by using the neural network is generally time-consuming in comparison with the well-studied conventional solution methods of differential equations and it does not seem useful to apply the neural network solution method to a usual problem for solving differential equations.

Our aims of the study are described as follows.

(1) Extension of the method by Lagaris et al., to a method with subdomain-defined form factors: As it is difficult to find a single appropriate form factor which describes the whole boundary/initial conditions we divide the domain into a number of subdomains and construct a set of a form factor in each subdomain. We apply the method to a simple model problem.

(2) Application of the neural network collocation method for data assimilation problems: One of the distinctive features of the neural network differential equation solver is that a smooth solution can be obtained even if experimetal/observational noise is included in constraining conditions such as initial/boundary conditions and so on. By making use of this feature the method can be applied efficiently to the data assimilation. We study basic problems concerning this application.

2. Neuralnet collocation method (NCM)

2.1 Analytical expression of derivatives of solution

In this subsection we consider a three-layered neural network. The result can be extended easily to a network with more layers. An analytical expression of a solution is given as

$$y_k = \sum_{j=1}^{J} w_{kj}^{(2)} \sigma\left(\sum_{i=1}^{I} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right)$$

where x, y, w, and σ are the input variable, the output variable, the weight variable, and the activation function (usually a sigmoid function: $\sigma(x) = 1/(1+e^{-x})$), respectively. w_{j0} is the offset.

By differentiating the solution with respect to the input variables a derivative of the solution are obtained as

$$\frac{\partial y_k}{\partial x_\ell} = \sum_{j=1}^{J} w_{kj}^{(2)} w_{j\ell}^{(1)} \sigma' \left(\sum_{i=1}^{I} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right)$$

$$\frac{\partial^2 y_k}{\partial x_\ell \partial x_m} = \sum_{j=1}^{J} w_{kj}^{(2)} w_{j\ell}^{(1)} w_{jm}^{(1)} \sigma' \left(\sum_{i=1}^{I} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right)$$

where $\sigma' = d\sigma(X)/dX$. By using these expression the residual of the differential equation is expressed analytically.

2.2 Collocation method

We consider the following differential equation.

$$D\vec{y} = \vec{g}(\vec{x}), \qquad \vec{x} \in \Omega$$

where D is an operator. We assume the output value of the neural network is an approximation of the solution of the differential equation, i.e.,

$$\vec{y} \approx \vec{f}(\vec{x}, \vec{p})$$

where $\vec{p}$ is the weights expressed as a vector. The object function is, therefore, expressed as

$$E = \int \left\{ D\vec{f}(\vec{x}, \vec{p}) - g(\vec{x}) \right\}^2 d\vec{x}$$

Then the computational domain Ω is discretized to a domain $\hat{\Omega}$ of the collocation points and the object function for the collocation method is obtained as

$$E = \sum_{pattern} F(\vec{f}(\vec{x}_{pattern}, \vec{p}))$$

$$F(\vec{f}(\vec{x}_{pattern}, \vec{p})) = \left\{ D\vec{f}(\vec{x}_{pattern}, \vec{p}) - \vec{g}(x_{pattern}) \right\}$$

$$\vec{x}_{pattern} \in \hat{\Omega} \subset \Omega$$

2.3 Initial/boundary conditions

The boundary conditions of the Dirichlet type are given as follows.

$$\vec{y}(\vec{x}_b) = \vec{y}_b, \quad \vec{x}_b \in \Gamma$$

where Γ is the boundary of the computational domain. Essentially the same discussion holds for other kinds of boundary conditions. Initial conditions are also treated similarly.

According to the Milligen's method the initial/boundary conditions are imposed by preparing the following object function.

$$\hat{E} = E + \lambda E_b$$

$$E_b = \sum_i \left(\vec{y}(\hat{\vec{x}}_{bi}) - \hat{\vec{y}}_{bi} \right)^2$$

It should be noted that in this expression the points where the constraining

conditions are given should not necessarily be placed on the computational boundary.

In the case of the Lagaris' method the solution of the differential equation is approximated as $\vec{f}(\vec{x},\vec{p})$ by using the output data of the neural network $\vec{f}_{NN}(\vec{x},\vec{p})$ as

$$\vec{f}(\vec{x},\vec{p}) = \vec{A}(\vec{x}) + \vec{H}\left(\vec{x}, \vec{f}_{NN}(\vec{x},\vec{p})\right)$$

$$\vec{A}(\vec{x}_b) = \vec{y}_b$$

$$\vec{H}\left(\vec{x}_b, \vec{f}_{NN}(\vec{x}_b,\vec{p})\right) = 0$$

For the Dirichlet boundary conditions $\vec{H}\left(\vec{x}_b, \vec{f}_{NN}(\vec{x}_b,\vec{p})\right)$ is expressed as a product of an appropriately chosen form factor and the output data of the neural network $\vec{f}_{NN}(\vec{x},\vec{p})$. In this case the form factor vanishes at the boundary.

3. Boundary condition assignment by divided form factors

In this section we describe the method to apply the Lagaris' method to problems with a complicated boundary shape. This is realized by dividing the whole computational domain into subdomains with simpler boundary shapes. We solve the Poisson equation in a square domain and in a T-shaped domain. The Lagaris' method cannot be applied directly to the problem in the T-shaped domain and we divide the T-shaped domain into 7 subdomains.

3.1 Method of a divided form factor

The conditions which the form factor g used for the Dirichlet type boundary condition should satisfy are summarized as 1) it vanishes on the boundary, 2) it is positive (or negative) -definite in the domain, 3) it should be sufficiently smooth, and 4) derivatives are easy to calculate. As far as these conditions are satisfied choice of the form factor is arbitrary. In the following we consider two examples, i.e., the case of the square domain and the case of the T-shaped domain.

(1) Example 1 (the square domain: $(-1,1)\times(-1,1)$) (Fig.2)

In this case the simplest form factor is expressed as

$$g(x,y) = (1+x)(1-x)(1+y)(1-y)$$

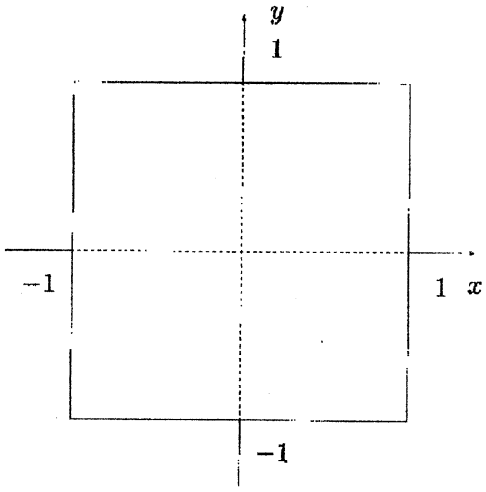


Fig.2 The square domain.

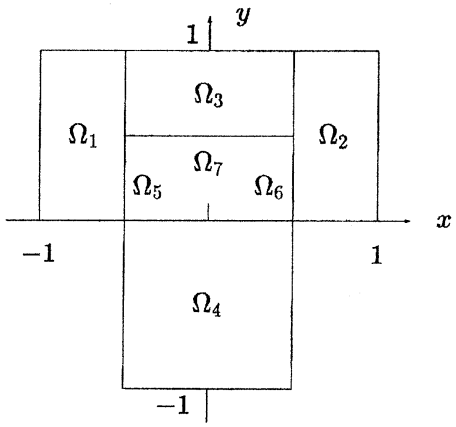


Fig.3 The T-shaped domain.

(2) Example 2 (the T-shaped domain: $(-1,1)\times(0,1)\cup(-1/2,1/2)\times(-1,1)$) (Fig.3)

For this case an example of the divided form factor is given as

$$g(x,y)=\begin{cases} x(1+x)y(1-y) & (\overline{\Omega}_1)\\ x(1-x)y(1-y) & (\overline{\Omega}_2)\\ y(1-y)/4 & (\overline{\Omega}_3)\\ (1/4-x^2)(1-y^2)/4 & (\overline{\Omega}_4)\\ r_1(1-r_1)/4 & (\overline{\Omega}_5)\\ r_1(1-r_1)/4 & (\overline{\Omega}_6)\\ 1/16 & (\overline{\Omega}_7) \end{cases}$$

$$\begin{pmatrix} r_1=\sqrt{(x+1/2)^2+y^2}\\ r_2=\sqrt{(x-1/2)^2+y^2} \end{pmatrix} \qquad (g: C^1 \ \ class)$$

3.2 Model problems

As a model equation we consider the following Poisson equation,

$$\Delta u=-\left(20x^3-\frac{15}{2}x\right)(y^3-y)-6\left(x^5-\frac{5}{4}x^3+\frac{1}{4}x\right)y \qquad \text{in } \Omega$$

$$u=0 \qquad \text{on } \Gamma$$

We solve the above equation for the two types of computational domains (Fig.4) described in the previous section. The exact solution is given for the above problems as

$$u(x,y)=(x-1)(x-\frac{1}{2})x(x+1)(x+\frac{1}{2})(y-1)y(y+1)$$

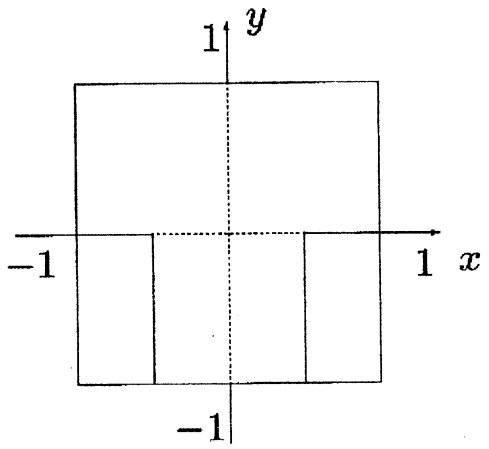
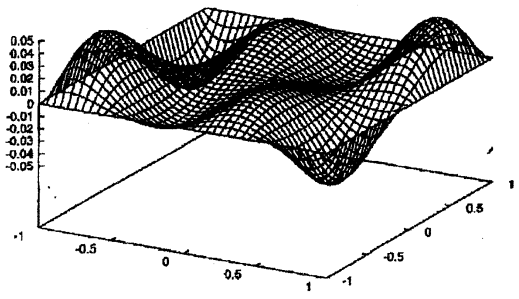


Fig.4 The computational domains for the model problems.

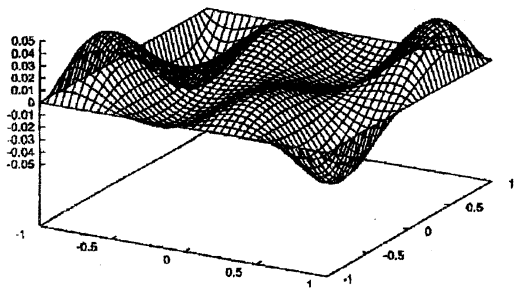
For simplicity of programing we employed a training algorithm composed of a random search and the linear least square method instead of the conventionally used error backpropagation method. The structure of the neural network is composed of two PUs in the input layer, 40 PUs in the hidden layer, and 1 PU in the output layer (we denote this structure as 2-40-1). The results of the calculations are shown in Figs.5-8 and the magnitude of the errors is summarized in Table 1, where the maximum random value denotes the maximum range of the initially assigned weights.

Table 1 Errors of solutions of the two model problems
(The maximum of the exact solution is 4.3E-02)

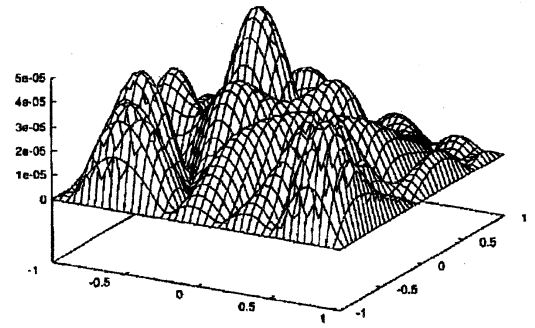
	Square domain	T-shaped domain
The maximum random value	1	10
RMS os residual	1.9E-03	6.4E-02
(Number of collocation points)	(117)	(89)
The maximum absolute error	5.7E-05	2.1E-03
(Number of evaluation points)	(1681)	(1281)



(a)

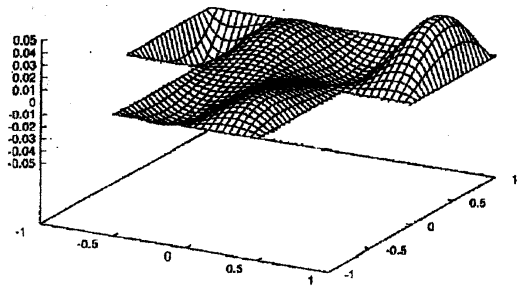


(b)

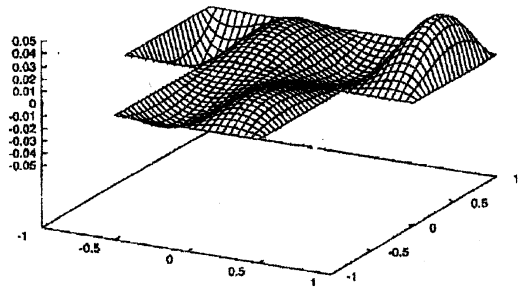


(c)

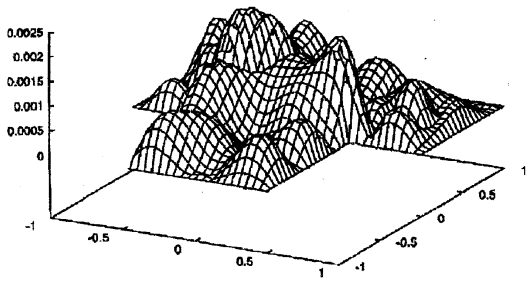
Fig.5 Birdeye view of the solution in the square domain
(a) NN collocation, (b) exact,
(c) absolute error



(a)

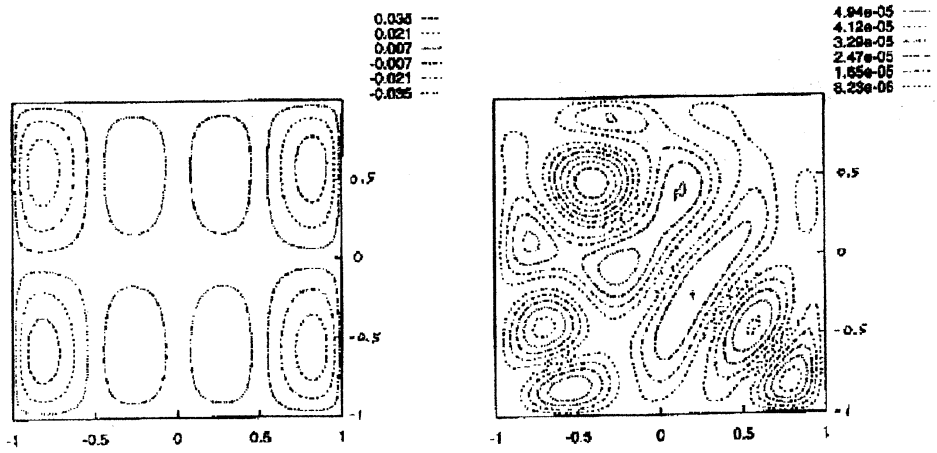


(b)



(c)

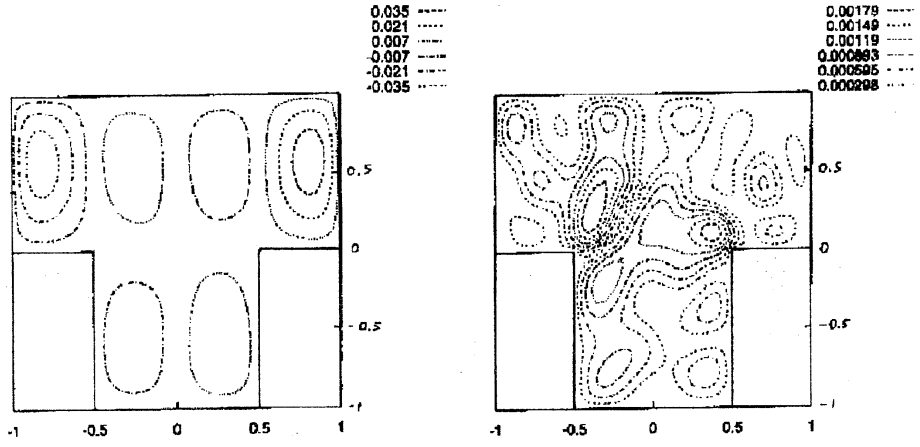
Fig.6 Birdeye view of the solution in T-shaped domain
(a) NN collocation, (b) exact.
(c) absolute error



Numerical solution

(a) (b)

Fig.7 Contour plot of the solution (a) in the square domain and the corresponding error.



Numerical solution Absolute error

(a) (b)

Fig.8 Contour plot of the solution (a) in the T-shaped domain and the corresponding error.

4. Application of NCM to a data assimilation problem

In this chapter we study basic issues of the neuralnet collocation method relating to the data assimilation problems.

4.1 Irregular boundary/initial conditions

In order to investigate the possibility to apply the neuralnet collocation method to the data assimilation problem we solve differential equations with constraining conditions given irregularly in space or time.

4.2 Solution of Lorenz equation

In this section we solve the Lorenz equation by assigning initial conditions at different temporal points, t_{0X}, t_{0Y}, and t_{0Z} for three variables, X, Y, and Z, respectively. For training the network we employed a combination of the error back-propagation method and the quasi-Newton method. The Lorenz equation is

$$\frac{dX}{dt} = \sigma(Y - X)$$

$$\frac{dY}{dt} = rX - Y - XZ$$

$$\frac{dZ}{dt} = XY - bZ$$

$$X(t_{0X}) = X_0,\ Y(t_{0Y}) = Y_0,\ Z(t_{0Z}) = Z_0$$

The object function of the neuralnet collocation method is given as follows.

$$E = \sum_{\hat{t} \in \hat{D}} \left[\begin{array}{l} \left(\frac{dX_t}{dt} + \sigma(X_t - Y_t) \right)^2 \\ + \left(\frac{dY_t}{dt} - rX_t + Y_t + X_t Z_t \right)^2 \\ + \left(\frac{dZ_t}{dt} - X_t Y_t + bZ_t \right)^2 \end{array} \right]_{t=\hat{t}}$$

$$\hat{D} \subset D \text{ and } |\hat{D}| < \infty$$

$$X_t = A + (t - t_{0X}) f_{NN}(t, \vec{p}_X)$$

$$Y_t = B + (t - t_{0Y}) f_{NN}(t, \vec{p}_Y)$$

$$Z_t = C + (t - t_{0Z}) f_{NN}(t, \vec{p}_Z)$$

$$t \in D,\ D = [0, 0.4], \qquad A = X(t_{0X}),\ B = Y(t_{0Y}),\ C = Z(t_{0Z})$$

At first, above equation is solved by the Runge-Kutta method for the following conditions to obtain a base solution for comparison of the neuralnet collocation method.

$$t \in D = [0, 0.4]$$

$$X_0 = 0.0,\ Y_0 = 1.0,\ Z_0 = 0.0, \qquad t_{0X} = t_{0Y} = t_{0Z} = t_0 = 0,0$$

$$\sigma = 1.0,\ r = 28.0,\ b = 8/3$$

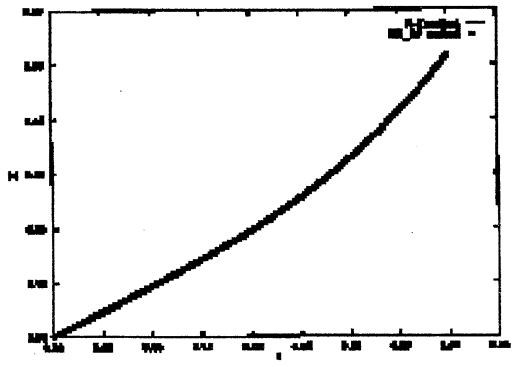
Next, we solve the same problem by the neuralnet collocation method imposing the initial condition at t=0. The results by the Runge-Kutta method and the neuralnet collocation method are shown in Fig.9, where they are indistinguishable each other in this scale. The error of the results is 2.22E-2 after the error back-propagation, and 2.16E-3 after the quasi-

Newton procedure.

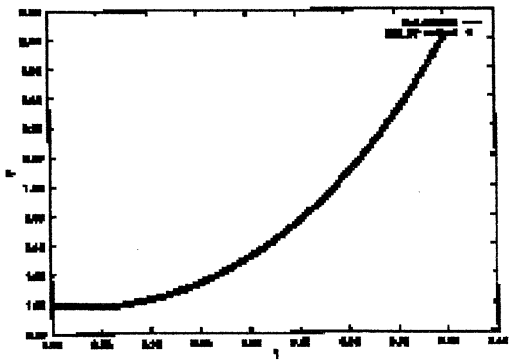
Then we solved the Lorenz equation for initial conditions imposed separately to different variables, X, Y, and Z. The initial conditions were obtained from the base solution by the Runge-Kutta method as

$$(X_0(t_{0X}), Y_0(t_{0Y}), Z_0(t_{0Z}))$$
$$= (0.1961(0.2), 1.0345(0.1), 0.0011(0.05))$$

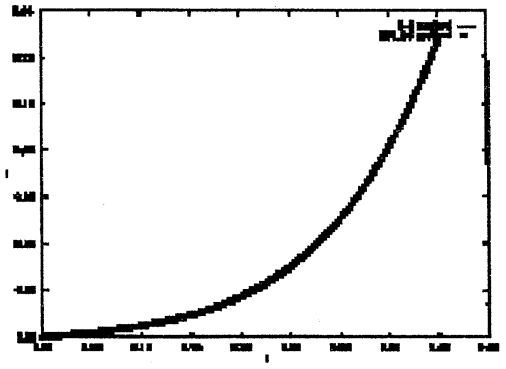
The computational results are shown in Fig.10 with the base solutions, where the results are also indistinguishable between them.



(a)

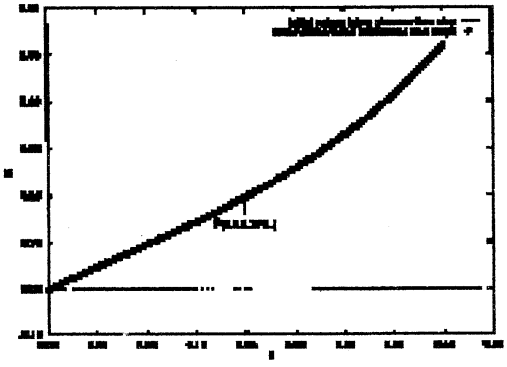


(b)

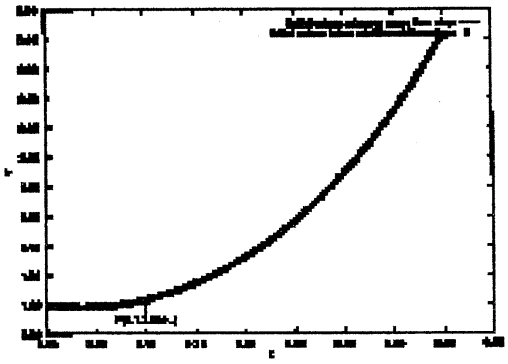


(c)

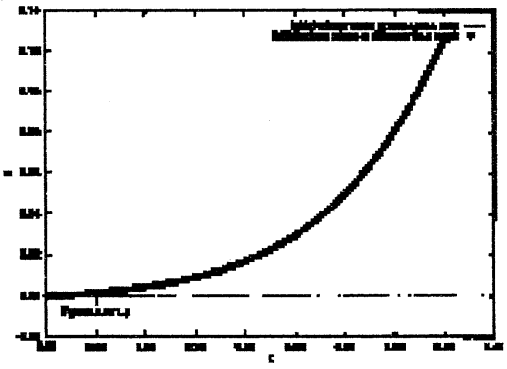
Fig. 9 Solutions of the Lorenz equation for the same initial conditions for X(a), Y(b), and Z(c).



(a)



(b)



(c)

Fig.10 Solutions of the Lorenz equation for separately given initial conditions.

4.3 Solution of a heat equation

In order to investigate a possibility to apply the neuralnet collocation method to the data assimilation problem we consider to solve the heat equation for irregularly imposed initial conditions.

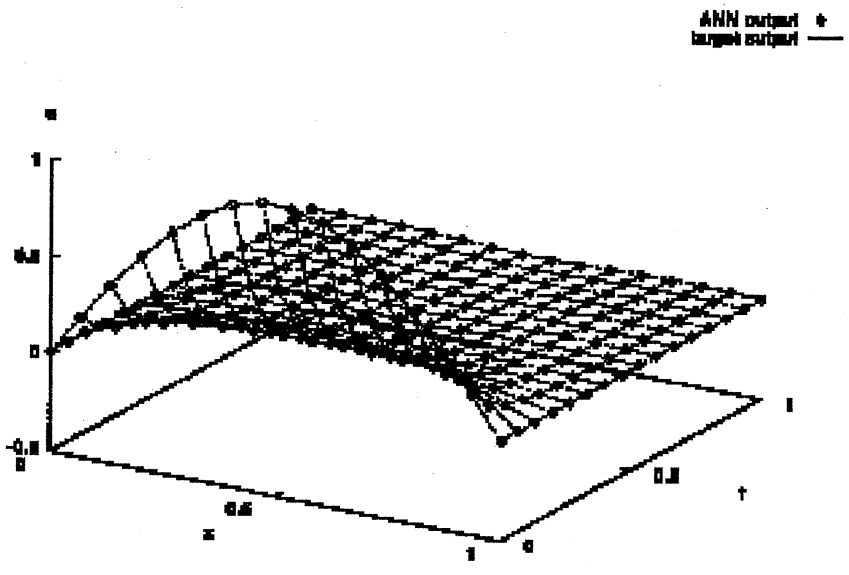
$$\alpha^2 \frac{\partial^2 u}{\partial x^2} + \sin 3\pi x - \frac{\partial u}{\partial t} = 0, \qquad (x,t) \in D = [0,1] \times [0.1]$$

$$\alpha = 1$$
$$u(0,t) = u(1,t) = 0.0$$
$$u(x,0) = \sin \pi x$$

The analytical solution to the above problem is given as

$$u(x,t) = \exp\left(-(\pi\alpha)^2 t\right)\sin \pi x + \frac{1 - \exp\left(-(3\pi\alpha)^2 t\right)}{(3\pi\alpha)^2}\sin 3\pi x$$

The same problem was solved by the neuralnet collocation method. The result is shown with the analytical solution in Fig. 11. Results with the average errors of 3.49E-3 and 1.33E-3 are obtained after 20000 iteration of the back-propagation method, and 5 iterations of the quasi-Newton method, respectively. By giving initial conditions for irregularly placed spatial points almost similar results were obtained.



5. Discussion and conclusion

(1) Extension of Lagaris' method to a case of subdomain-defined form factor

for the initial/boundary conditions is successfully applied to solution of Poisson equation in a square domain and in a T-shaped domain. To attain higher accuracy some improvements are necessary.

(2) In order to study a possibility to apply the neuralnet collocation method for the data assimilation problem simple model problems on the Lorentz equation and the heat equation were solved. Promising results were obtained.

References

[1] B.Ph. van Milligen, et al., Phys. Rev. Letters **75** (1995) 3594 - 3597.

[2] I.E. Lagaris, et al., Comput. Phys. Commun. **104** (1997) 1 - 14